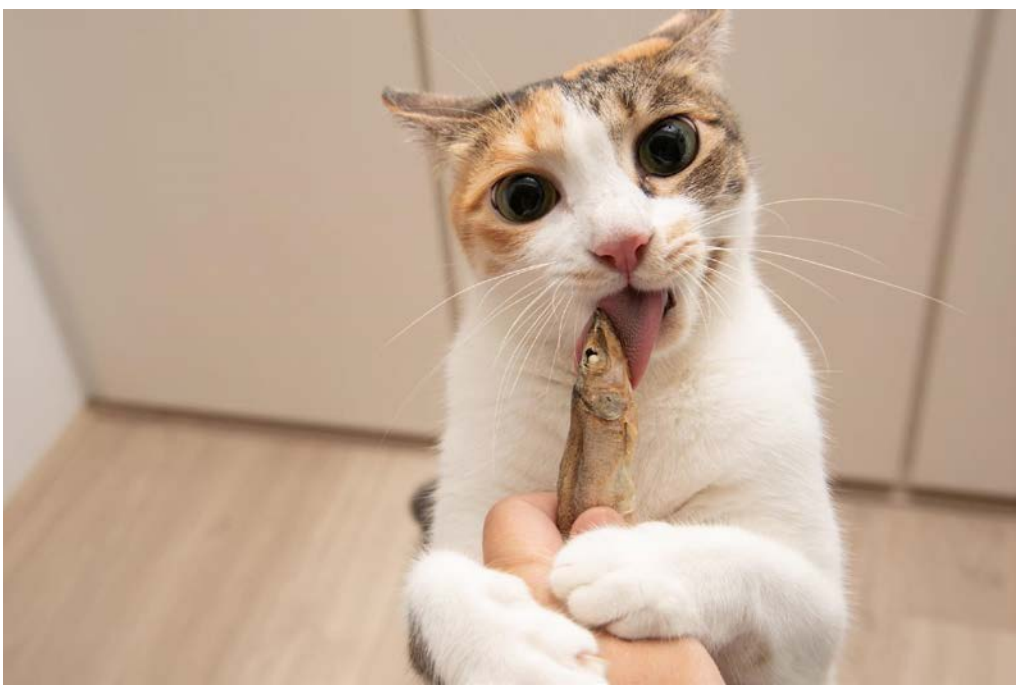
408410056 許庭涵 hw13

因為不確定原始程式碼是要放上第幾題的結果
(2、3程式碼內容不太一樣)
因此交上兩個程式碼分別為
fileCount1&2.c(第1&2題)、
fileCount3.c(第3題)



printf(“希望期末好運貓貓可以帶來好運”);

第一題計算資料夾內的字數，
使用pipe將ls接上stdout，wc接上stdin，最後輸出字數，相等於ls -R /YYY | wc -l
部分程式碼如下:

```
pipe(pipefd);
pid1 = fork();   //產生第一個child
if (pid1==0) {
    setpgid(0, 0);
    close(1);   //關閉stdout
    dup(pipefd[1]); //將pipefd[1]複製到stdout
    close(pipefd[1]);   //將沒用到的關閉
    close(pipefd[0]);   //將沒用到的關閉
    execlp("ls", "ls" ,"-R", argv[1], NULL);   //執行ls，ls會將東西藉由stdout輸出到pipefd[1]
}
if (pid1>0) {
    pid2 = fork();//產生第二個child
    if (pid2==0) {
        setpgid(0, pid1);
        close(0);   //關閉stdin
        dup(pipefd[0]); //將pipefd[0]複製到stdin
        close(pipefd[1]);   //將沒用到的關閉
        close(pipefd[0]);   //將沒用到的關閉
        execlp("wc","wc","-l", NULL);   //執行wc，wc將透過stdin從pipefd[0]讀入資料
    }
}
```

第二題按下^C後要立即中止

```
setpgid(0, 0);
setpgid(0, pid1);
```

∵這兩行會將child新增至新的群組(child2會加入child1群組)
∴因此我們要針對^C做一些變化

```
signal(SIGINT, signal_ctr_c);
```

<<放入對SIGINT的函式

```
void signal_ctr_c(int signum) {
    //殺掉process group
        fprintf(stderr, "kill process group %d\n", -1*pid1);
        kill(-1*pid1, signum);
    //parent結束離開
    _exit(0);
}
```

按下^C會跳到這個函式執行，將child的grp殺掉，完成按下^C即可parent連child一起結束的功能

第三題要求不放入signal函式，也就是不修改原本linux對SIGINT的詮釋

不放入這兩行>>

```
setpgid(0, 0);
setpgid(0, pid1);
```

也就是不為child新增群組，那parent跟child都會在同一群組
因此不用新憎signal函式特別處理^C就可以將程式立即中止

參考資料:

論壇>>關於control-c 和 process control group的一些問題

3. 在我的 fork.c 中，只要 parent 和 child 都存在，按下 ^C 會把兩個程式都殺掉；但是在 pipe4-3.c 中，註解掉 signal 的情形下，同樣 parent 和 child 都存在，但是按下 ^C 不會把兩個程式都殺掉，只有 parent 被殺掉，ls, sort 兩個 child 都還在執行，怎麼會這樣？(這個程式的 signal handler 也只是利用 kill function 達到殺掉 child 的效果而已)

按下 ^C時，terminal會送出SIGINT給foreground process group。如果parent和child都隸屬於foreground process group，那麼parent和child會一起死掉
但pipe4-3.c 會產生新的process group，control-c只會送給foreground process（即：parent）因此只有parent會死掉，child不會死掉